IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s)	: Stephan Courcambeck et al	Confirmation No.	: 1111
Application No.	: 10/817325	Group Art Unit	: 2136
Filing Date	: April 1, 2004	Examiner	: Fikremariam A. Yalew
Docket No.	: 2269-019-03	Customer No.	: 00996
Title	: PROTECTION OF A PROGRAM WAITING TO BE EXECUTED IN A MEMORY USED BY A MICROPROCESSOR		

PRE-APPEAL BRIEF

Applicant's agent requests the Review Panel to reconsider the application in view of this communication. Claims 1-5 and 18-24 are pending. Claims 1-5 and 18-24 are rejected.

Rejection of claims 1-5 and 18-24 under 35 U.S.C. § 103(a) Over Sinha et al. (U.S. Patent No. 7,346,780) in view of Douceur et al (U.S. Pub. No. 2005/0132375)

Claim 1

Claim 1 recites, calculating, on each task change between a first program module switching from foreground to background and a second program module switching from background to foreground, a signature of at least part of the second program module instruction lines.

For example, according to an embodiment, a CPU may have the capability of multitasking between executing two or more separate and distinct program modules. As one program module is executed and engaged by the CPU, it may be referred to as running in the foreground. Other program modules that are executing, but are not currently engaged by the CPU in the execution of specific tasks may be referred to as running in the background. Thus, in an effort to ensure that a background program module has not been tampered with by malicious software of hackers, each time a

program module transitions to become a foreground application in which the CPU is executing instructions from said application, at least part of the application that is switching from the background to the foreground is used to generate a signature. This signature may be compared to a stored signature that was generated and stored in a memory when the application transitioned to the background. If the signatures match, then this match may be interpreted to mean that the program module now transitioning back to the foreground has not been changed or modified. See generally, paragraph [8] of the detailed description section of the specification.

However, *Sinha et al* teaches a method that has several differences that ultimately teach away from the recitations of claim 1. In specific, *Sinha et al* is directed, generally, toward a digital rights management (DRM) solution that may verify that two instances of the same program module are unaltered duplicates of each other, (*e.g.*, typically when operating on separate computer systems) using a so-called "integrity veracitication" technique as generally described in the cited and applied section, column 7, line 31 to column 8, line 25 of *Sinha et al*.

The Examiner correctly acknowledges that *Sinha et al* offers no teaching or discussion about performing such a signature calculation in response to a CPU switching tasks between two different and simultaneously executing programs (*i.e.*, switching an application from background to foreground and vice versa). Moreover, there is no teaching anywhere in *Sinha et al* with regard to foreground and background execution in any capacity and the whole of *Sinha et al* is silent with respect to even the concept of background and foreground execution.

Douceur et al does not remedy this deficient teaching. *Douceur et al* is directed to a system and method for limiting the effects of background tasks on executing foreground tasks as the CPU is a limited resource. Thus, the system and method ensures that any background tasks that are executing at a "groveler" (*e.g.*, a task engine dedicated solely to background tasks) does not interfere with the more important foreground tasks currently executing. Further, the cited paragraphs [0005]-[0006]. [0036], and [00364]-[0065] describe a feature that solely part of the background task engine wherein files may be checked against memory to see if anything has changed since the background task engine was operating on a particular file. This is because the background task engine may be suspended for large amounts of time if the

foreground tasks heavily utilize the CPU. There is simply no teaching anywhere in *Douceur et al* that teaches or even suggests the possibility of tasks in the "groveler" being executed by a different task engine – namely the foreground task engine. Thus, no prior art of record, whether considered individually or in any permissible combination with each other, teaches program modules switching between background execution and foreground execution.

Inexplicably, the Examiner has argued that a motivation to combine these teachings of *Sinha et al* and *Douceur et al* is to attain a resulting "system for limiting the interference of background process on the foreground process." Such a motivation to combine these references in this manner has absolutely nothing to do with that which is recited in claim 1. Specifically, checking signatures of tasks switching from background execution to foreground execution makes irrelevant any impact simultaneously executing tasks may have on each other. Such a twist in logic in a clear indication of hindsight reasoning wherein the Examiner is finding motivation to solve problems that do not exist. As a matter of law, obviousness may not be established using hindsight obtained in view of the teachings or suggestions of the applicants. To guard against the use of such impermissible hindsight, obviousness needs to be determined by ascertaining whether the applicable prior art contains any suggestion or motivation for making the modifications in the design of the prior art article in order to produce the claimed design and not a made-up problem. The mere possibility that a prior art teaching could be modified or combined such that its use would lead to the particular limitations recited in a claim does not make the recited limitation obvious, unless the prior art suggests the desirability of such a modification.

Further, as mentioned above, *Sinha et al* actually teaches away from the recitations of claim 1 because *Sinha et al* is only concerned with a single serial application execution wherein any calculation is only performed when the program module is first executed. Therefore, there is no possible way to check the program module once it has been executed without first shutting it down again and restarting. Such a limitation is hardly comforting when malicious software is designed to attack currently executing software as is the goal of the recitations of claim 1. Any conclusion that these two references could be combined to accomplish a security task that neither one is concerned with is broad and conclusory. Such broad, conclusory statements do

not come close to adequately addressing the issue of motivation to combine, are not evidence of obviousness, and therefore are improper as a matter of law. *In re Dembiczak*, 175 F.3d 994, 999, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

Claims 2-3

Claims 2-3 are allowable by virtue of their dependence from claim 1, and for at least the reasons given for claim 1.

Claim 4

Claim 4 recites a processor of multitask execution of several programs, each of the several programs being different from each other exploiting a table of correspondence, each correspondence being associated with an identifier of the involved program, comprising means for calculating a current signature, and means for comparing this signature with the identifier of the program stored in the correspondence table. There is simply no disclosure anywhere in *Sinha et al* or in *Douceur et al* that can be construed to teach exploiting a table of correspondence, each correspondence being associated with an identifier of the involved program. Thus, no permissible combination of *Sinha et al* and *Douceur et al* teaches or suggests all of the elements and features of this claim.

Claims 5

Claims 5 are allowable by virtue of their dependence from claim 4 and for at least the reasons given for claim 4.

Claim 18

Claim 18 recites, executing, at a CPU, a plurality of programs simultaneously, each program having a unique signature calculated when first executed and calculating, on each task change, a new signature, and checking the conformity of the new signature with the unique signature. The Examiner acknowledges that *Sinha et al* does not teach calculating, on each task change, a new signature, and checking the conformity of the new signature with the unique signature.

Douceur et al does not remedy this deficient teaching. *Douceur et al* teaches, a feature that solely part of the background task engine wherein files may be checked against memory to see if anything has changed since the background task engine was operating on a particular file. This is because the background task engine may be suspended for large amounts of time if the foreground tasks heavily utilize the CPU. Thus, Douceur et al does not teach tasks that change.

Further, *Sinha et al* actually teaches away from the recitations of claim 18 because *Sinha et al* is only concerned with a single serial application execution wherein any calculation is only performed when the program module is first executed. Any conclusion that these two references could be combined to accomplish a security task that neither one is concerned with is broad and conclusory. Such broad, conclusory statements do not come close to adequately addressing the issue of motivation to combine, are not evidence of obviousness, and therefore are improper as a matter of law. Further, as discussed above, the Examiner is clearly using hindsight reasoning which is impermissible as a matter of law.

Claims 19-24

Claims 19-24 are allowable by virtue of their dependence from claim 18 and for at least the reasons given for claim 18.

For reasons described above, the claims are now in condition for allowance, which is earnestly solicited.

DATED this 3rd day of February, 2011.

Respectfully submitted,

/Kevin D Jablonski/

Kevin D. Jablonski
Registration No. 50401

Customer No. 00996

Graybeal Jackson Haley LLP
400 - 108th Avenue NE, Suite 700
Bellevue, Washington 98004
Telephone: 425.455.5575
Facsimile: 425.455.1046